AF/2193
IFW

PTO/SB/21 (09-04)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Total Number of Pages in This Submission

Application Number	09/872,413
Filing Date	6/1/2001
First Named Inventor	Pastor
Art Unit	2193
Examiner Name	Chavis
Attorney Docket Number	CHG-001.1P

ENCLOSURES *(Check all that apply)*

- [x] Fee Transmittal Form
  - [x] Fee Attached
- [ ] Amendment/Reply
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Reply to Missing Parts/ Incomplete Application
  - [ ] Reply to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) _____
  - [ ] Landscape Table on CD
- [ ] After Allowance Communication to TC
- [x] Appeal Communication to Board of Appeals and Interferences
- [ ] Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- [ ] Proprietary Information
- [ ] Status Letter
- [x] Other Enclosure(s) (please Identify below):
  Return receipt postcard, Appeal Brief & Credit Card Auth Form for $250

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	Ronald Craig Fish, A Law Corporation		
Signature			
Printed name	Ronald C. Fish		
Date		Reg. No.	28,843

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

Signature	Ronald C. Fish		
Typed or printed name	Ronald Craig Fish	Date	5/11/06

PTO/SB/17 (12-04v2)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Effective on 12/08/2004.
Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).

FEE TRANSMITTAL
For FY 2005

☑ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT	($)	250.00

Complete if Known	
Application Number	09/872,413
Filing Date	6/1/2001
First Named Inventor	Pastor et al.
Examiner Name	Chavis
Art Unit	2193
Attorney Docket No.	CHG-001.1P

METHOD OF PAYMENT (check all that apply)

☐ Check ☑ Credit Card ☐ Money Order ☐ None ☐ Other (please identify): _____

☑ Deposit Account Deposit Account Number: 50-3592 Deposit Account Name: RONALD CRAIG FISH, LAW

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☐ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, **except for the filing fee**

☑ Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 ☑ Credit any overpayments

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

FEE CALCULATION

1. BASIC FILING, SEARCH, AND EXAMINATION FEES

	FILING FEES		SEARCH FEES		EXAMINATION FEES		
		Small Entity		Small Entity		Small Entity	
Application Type	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fees Paid ($)
Utility	300	150	500	250	200	100	
Design	200	100	100	50	130	65	
Plant	200	100	300	150	160	80	
Reissue	300	150	500	250	600	300	
Provisional	200	100	0	0	0	0	

2. EXCESS CLAIM FEES

Fee Description	Fee ($)	Small Entity Fee ($)
Each claim over 20 (including Reissues)	50	25
Each independent claim over 3 (including Reissues)	200	100
Multiple dependent claims	360	180

Total Claims		Extra Claims	Fee ($)		Fee Paid ($)
49	- 20 or HP =	0	x 25	=	0

HP = highest number of total claims paid for, if greater than 20.

Indep. Claims		Extra Claims	Fee ($)		Fee Paid ($)
12	- 3 or HP =	0	x 100	=	0

HP = highest number of independent claims paid for, if greater than 3.

Multiple Dependent Claims	
Fee ($)	Fee Paid ($)

3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper (excluding electronically filed sequence or computer listings under 37 CFR 1.52(e)), the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets		Extra Sheets		Number of each additional 50 or fraction thereof	Fee ($)		Fee Paid ($)
100	- 100 =	0	/ 50 =	0 (round up to a whole number)	x 125	=	0

4. OTHER FEE(S)

	Fees Paid ($)
Non-English Specification, $130 fee (no small entity discount)	
Other (e.g., late filing surcharge): appeal brief fee per 41.20(b)(2)	

SUBMITTED BY

Signature	Ronald C. Fish	Registration No. (Attorney/Agent) 28,843	Telephone 408 866 4777
Name (Print/Type)	RONALD CRAIG FISH		Date 5/11/06

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Pastor et al.

Art Unit: 2193

Examiner: Chavis

Serial No. 09/872,413

Docket No. CHG-001.1P

Filed: 6/1/2001

For: AUTOMATIC SOFTWARE PRODUCTION SYSTEM

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, Va. 22313-1450

May 11, 2006

APPLICANT'S APPEAL BRIEF

REAL PARTY IN INTEREST

The real party in interest is Constructiones Hispano Germanas, S.A. the parent company of Care Technologies, S.A. and Sosy, Inc.

RELATED APPEALS AND INTERFERENCES

None

STATUS OF CLAIMS

Claims 25-73 are pending. Claim 25 is finally rejected, and claims 26-73 have been withdrawn from consideration by Examiner action in *sua sponte* applying 37 CFR 1.142(b) and incorrectly holding claims 26-73 are directed to a non-elected invention.

STATUS OF AMENDMENTS

The amendment dated 10/7/2005 canceling original claims 1-24 and adding new claims 25-73 has been entered, and the final rejection was mailed 1/27/06. There have been no amendments after the final rejection.

SUMMARY OF CLAIMED SUBJECT MATTER

References to page and line numbers of the specification supporting the independent claims discussed below will be given in the form [X/YY] where X is the page number and Y Y is the line number on that page.

The invention of independent claim 25 is the modeler subsystem of an automated software production system. The modeler subsystem controls a computer to receive user input data that defines the primitives of a conceptual model (of a computer program to be written by the automated software production system (the target application), automatically converts that user entered data into a formal language specification and validates the formal language specification to ensure it is complete, correct and unambiguous.

Specifically, claim 25 is the following, and supporting passages in the specification are included in parenthesis:

> 25. [NEW] A computer-readable medium having stored thereon computer-readable instructions, which, when executed by a computer, cause said computer to perform the following process:
>
> receive user input data [7/17-21 and 32/9-10] that defines primitives [7/21] of a conceptual model [5/2], said conceptual model being comprised of an object model [21/21 et seq], a dynamic model [24/11 et seq.], a functional model [26/16] and a presentation model [29/6] which together define the complete functionality of a target computer program to be automatically

generated [6/22] and which defines complete interface mechanisms [29/11-20] for interaction of user or other processes with the functionality of said target computer program [29/11-12], no other software code, code component, code libraries or any other third party software artifact being necessary [6/22] to completely define the functionality of said target computer program and its interface mechanisms;

automatically converting [32/24] said user input data into data structures [32/30 and 9/16] in the form of formal language statements [5/2] organized by a formal language syntax [rules that can be used for validation - 5/3 and 5/12 and 5/27], the collection of all such formal language statements [5/8] forming a formal language specification [5/11 and 32/10];

validating [5/3 and 5/17 and 40/22 et seq.] said formal language specification to ensure it is complete [40/28-29], correct [41/1-2] and unambiguous [5/11 and 41/1-2]] to prepare said formal language specification for automatic translation [6/22-23] into complete, operative code of said target computer program which is the functional equivalent of said conceptual model and having a user interface defined by the primitives [29/18-20 primitives of the presentation model are the presentation patterns] entered by said user which define said presentation model [29/6-12].

The primitives of a conceptual model define the complete desired functionality of the target application and include data structures that form part of an object model, a dynamic model, a functional model and a presentation model. Each of these models is a component of the overall conceptual model.

The object model defines the classes of objects in the target application and the class framework where the class signature is precisely declared. For each class, there are primitives that define a set of constant, variable and derived attributes,

primitives that define a set of services including private and shared events and local transactions, primitives that define a set of integrity constraints specified for the class; and primitives that define derivation expressions which give values to the derived attributes.

The dynamic model defines the valid lives of objects from creation to destruction and defines service preconditions in the form of formulas which label state transitions and defines trigger relationships and global transactions. The dynamic model uses two kinds of diagrams. A state transition diagram has primitives which define service preconditions in terms of formulas labeling the state transitions and primitives which define the process definition of the class where the template for valid object lives is fixed. From the object interaction diagram, primitives containing two other features of the formal language specification are resident: trigger relationships and global transactions.

The functional model contains as primitives the dynamic formulas related to evaluations where the effect of events on attribute values is specified.

The presentation model contains primitives which define patterns for a user interface of the target application. In the claims, these primitives are referred to as the various types of patterns which a user specifies to specify the type of interface desired for the target application.

Independent claim 37 is as follows and support for the various limitations is included in square brackets in the claim.

37. [NEW] A computer-readable medium having stored thereon a data structure comprising:

a collection of fields [repository 32/19-28] that store data entered

by a user [Figures 9a, 9C, 10, 11A and 11B, 12, 13, 14, 15, 16, 17, 18, 19, 20], defining a conceptual model [5/2] of a computer program that the user wants automatically written by a translator process running on a computer, said conceptual model comprised of:

an object model data structure [5/3] which defines a system class architecture [Figure 10] comprised of one or more classes of objects defined by user input [Figures 9A, B and C], each class having attributes [Figures 3 and 16; 21/31-32] the value of which collectively define the state [9/12] of a system defined by said conceptual model (hereafter the target program);

a dynamic model data structure [5/3] which defines valid object life cycles [24/13] and which interobject communications can be established and defined by user input [24/13-14];

a functional model data structure [5/3 and 26/16 et seq.] which defines the semantics associated with any change of an object state as a consequence of an event occurrence [26/25-28], user input defining data structures which define the class and which variable attribute of said class is affected by which event of that class and how the event affects the value of said variable attribute [Fig. 15 and 26/27-30 and 27/8-11], said user input also defining a mathematical or logical valuation that defines how said variable attribute's value will be changed when said event happens [27/10];

a presentation model data structure [29/6 et seq.] defined by user input [Figure 19 and 20] and which defines a full user interface defining how users or other processes will be able to interact with the functionality of the target program, said user input defining a set of patterns [29/16-20] that specify which services

will be available to user of the system [Figure 12] and which information about the state of the target program users of the target program will be able to query [Figure 20].

Independent claim 39 and its support in the specification is:

39. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer cause said computer to carry out the following process:

soliciting and receiving user input [Figures 9(A) to 20] and converting said user input to data structures [5/2-3] which define primitives [7/21] of a conceptual model [5/2] comprising an object model [21/21 et seq] comprising one or more classes of objects which have attributes the state of which collective define the state of the system [86/6], and a dynamic model [24/11 et seq.], and a functional model [26/16] and a presentation model [29/6];

automatically converting [32/24] said data structures [32/30 and 9/16] of said conceptual model into statements [5/2] according to the syntax and semantics of a formal language [32/25-28], said collection of formal language statements defining a formal language specification [10] which defines the functionality of a computer program to be automatically written [32/9-10] (hereafter the target program) by a computer executing a translator process, said target program being a full computer program having a user interface specified by data structures comprising said presentation model [29/11-20], and not a prototype which needs additional code, code components, code libraries or any third party software artifact and which will be the full functional equivalent of said conceptual model [6/22];

and wherein said data structures define the functionality [26/16 et

seq.] of said conceptual model in terms of services which change the state of a system implemented by said target program [Figure 15], said services being events [38/30] or local or global transactions [38/31-32 and 63/5], and wherein events are defined as the smallest execution units possible in the scope of a class within said object model in that it is not possible to decompose an event into more elementary execution units [38/30], and wherein said local or global transactions are molecular execution units that have their functionality defined in terms of a sequence and/or alternation of other services each of which can be either an event or transaction [38/31-32 and 63/5, and wherein the functionality of a local or global transaction is explicitly defined by a data structure [19/13-14] which defines a formula [20/6-9 and 26/28-30] which expresses which services the transaction encompasses and the value assigned to every argument of every service comprising the transaction , and wherein said data structures which define local transactions limit the effect of a local transaction to affecting the state of primarily an instance of the class owning said transaction and potentially instances of classes related with the class owning said transaction [39/29-31 and Figure 13], and wherein global transactions are not limited to affecting the state of only one instance of a class and can affect the state of any instance or set of instances of any class or set of classes [39/9-11];

and wherein said solicited user input includes user input which defines valuations for said functional model [26/28-30], where each valuation is a primitive relating a variable attribute and an event of a class [Id.] and defines how the occurrence of said event affects the value of said variable attribute thereby explicitly defining the functionality of said event [Id.].

Independent claim 44 and its support in the specification is as follows:

44. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer cause said computer to perform the following process:

receive user input [Figure 9A] which defines a creation event [45/29 and 20/4-5] an optional destruction event [24/31 to 25/1 and 20/4-5] and an optional set of modification events [implemented by valuations --Figure 15, 26/28-30 and 19/25-27 and 63/23/26] for a class which is to become part of an object model of a conceptual model of a computer program to be automatically written, said user input defining said creation event creating an instance of a class owning said creation event and providing a value for all constant and variable attributes of the class which are required upon creation [Figure 9A dialog allows user to name class, give it its alias or identifier and creates a creation event; see also 41/23 as to what user input is required to create a class instance] , said user input also establishing all required relationships with instances of classes related with the instance of the class owning said creation event [Figure 9C and 54/4-7], and wherein said user input defines a destruction event [24/31 to 25/1 and 35/4-6 and 20/4-5] which eliminates all relationships between a given instance of a class owning said destruction event and related instances of classes which have relationships with said class and then destroying the instance of the class owning said destruction event [24/31 to 25/1 and 35/4-6 and 20/4-5], and wherein said user input defines a set of modification events each of which uses user input to modify the values of one or more variable attributes of a given instance of the class owning said modification event , the effect of said modification defined by one or more logical or mathematical valuations defined by user input and which affect the value of one or more variable attributes upon the occurrence of said modification event [implemented by valuations --Figure 15, 26/28-30 and 19/25-

27 and 63/23/26];

converting said user input into data structures [5/2-3 and 32/30 and 9/16] which define an object model having a class having said creation, destruction and modification events.

Independent claim 47 and its support in the specification is as follows:

47. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer cause said computer to perform the following process:

receiving user input [Figure 15] to define a valuation which will affect the value of a variable attribute of a class upon the occurrence of an event [27/10] and the satisfaction of an optional condition [27/2-4, 10], said solicited user input also defining which variable attribute of which class will be affected and the event which will trigger said valuation to affect the value of said variable attribute [Figure 15 and 27/1-217; and

converting said user input into data structures which define said valuation as part of a conceptual model of a computer program to be automatically written [27/27-31].

Independent claim 57 and its support in the specification is as follows:

57. [NEW] A computer-readable medium having computer-readable instructions stored thereon, which, when executed by a computer, control said computer to solicit user input [Figure 15 dialog box] to define a transaction [38/31-32 and 39/1-4 and 63/5-8],wherein said transaction is a molecular execution unit [38/31] expressed in terms of a formula [20/6-9 and 26/28-30] that specifies services in the form of events or transactions [20/1-2] which together comprise said molecular execution unit, and said computer-readable instructions controlling said computer to convert said user input [Figure 13] into

one or more data structures [5/1-3] which implement said transaction [19/13-14 and 39/29-31 and Figure 13].

Independent claim 64 and its support in the specification is as follows:

64. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer control said computer to perform the following process:

soliciting user input [7/17-21 and 32/9-10] to define primitives [7/21] of a conceptual model [5/2] defining the functionality of a target computer program to be written automatically, said conceptual model comprised of an object model [21/21 et seq], a dynamic model [24/11 et seq.], a functional model [26/16] and a presentation model [29/6];

converting said user input [32/24] into data structures [32/30 and 9/16] which implement said conceptual model in the form of statements in a formal language [5/2 and 5/8 - 9] syntax [rules that can be used for validation - 5/3 and 5/12 and 5/27] that together comprise a formal language specification [5/11 and 32/10], said formal language specification defining a complete computer program [6/18] having a user interface [29/11-20] and which needs no additional code, code components, code libraries or any other third party software artifact in order to be functionally equivalent to the conceptual model [18/28-29];

and wherein said data structures and formal language specification define the functionality of the conceptual model [26/16 et seq. in terms of how the state of the system is changed by means of services [Figure 15 and 26/28-30], said services being events [38/30] and transactions where transactions are comprised of other services and falling in either a local or global [19/14] category [20/6-7 and 38/31-32 and 63/5];

and wherein said data structures and formal language specifications

define said presentation model [29/6] in the form of user interface mechanisms [29/16-20] which define how a user or other processes will be able to interact with the functionality of the system [Figure 12].

Independent claim 67 and its support in the specification is as follows:

67. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer control said computer to perform the following process:

soliciting user input [7/17-21 and 32/9-10] to define primitives [7/21] of a presentation model [29/6] which is part of a conceptual model of a target computer program to be automatically written, said presentation model defining a user interface by a set of patterns [29/16-20] which will be used to define how users can interact with said target computer program by specifying the interaction scenarios for services which will be available to users of the target computer program [Figure 12] and the interaction scenarios to query information about the state of the target computer program which users of the system will be able to query [26/6 and 49/23-27 and 69/3-7];

and converting said user input to one or more data structures [32/30 and 9/16] which define said patterns of said presentation model.

Independent claim 70 and its support in the specification are as follows:

70. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer, cause said computer to carry out the following process:

solicit user input [7/17-21 and 32/9-10] which defines a presentation model [29/6] which defines the user interface for a target computer program [6/22] defined by a conceptual model [5/2], said solicited user input defining at least a service presentation pattern [29/18] for every service of the system specifying how user of the system will be able to invoke a service of said

system [29/21-24] and said solicited user input further defining:

an optional introduction pattern [29/25-30] assigned to every input data valuated argument of a service to which said service presentation pattern is assigned;

an optional defined selection pattern [30/1-8] assigned to every input data valuated argument of the service to which said service presentation pattern is assigned;

an optional population selection pattern [30/9-16] assigned to every input object valuated argument of a service to which said service presentation pattern is assigned;

an optional dependency pattern [30/17-19] assigned to every input argument of a service to which said service presentation pattern is assigned; and

converting said solicited user input into one or more data structures [5/3 and 32/30 and 9/16] which implement said service presentation pattern.

Independent claim 71 and its support in the specification is:

71. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer, cause said computer to carry out the following process:

solicit user input [7/17-21 and 32/9-10] which defines a presentation model [29/6-12] which defines the user interface [29/18-20 primitives of the presentation model are the presentation patterns] for a target computer program defined by a conceptual model [5/2], said solicited user input defining at least an instance presentation pattern [31/4-7] for every class of the system specifying how users of said target computer program will be able to query the state of a given instance of a class of said target computer program, said solicited user input further comprising user input which defines:

a display set pattern [19/17 and 20/20 - 22];

a set of available services of the class owning said instance presentation pattern [19/16-18];

a set of navigations to presentation patterns owned by the classes related to a class owning said instance presentation pattern [31/4-7]; and

converting said user input into one or more data structures [5/3] implementing said instance presentation pattern.

Independent claim 72 and its support in the specification are as follows:

72. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer, cause said computer to carry out the following process:

solicit user input [7/17-21 and 32/9-10] which defines a presentation model [29/6-12] which defines the user interface [29/18-20 primitives of the presentation model are the presentation patterns] for a target computer program defined by a conceptual model [5/2], said solicited user input defining at least a class population presentation pattern [31/8-11] for every class of the system specifying how users of said target computer program will be able to query the population of a given class of said target computer program [31/8-11], said solicited user input further comprising user input which defines:

a display set pattern [19/16-17 and 32/1-3];

an optional filter pattern [19/16-17];

an optional order criterion pattern [30/9-11 and 32/4-7];

a set of available services [19/16-18] of the class owning said class population presentation pattern;

a set of navigations to presentation patterns owned by the classes related to a class owning said class population presentation pattern [78/8-10];

and

converting said user input into one or more data structures [5/3] implementing said class population presentation pattern.

Independent claim 73 and its support in the specification is as follows:

73. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer, cause said computer to carry out the following process:

solicit user input [7/17-21 and 32/9-10] which defines a presentation model [29/6-12] which defines the user interface [29/18-20 primitives of the presentation model are the presentation patterns] for a target computer program defined by a conceptual model[5/2], said solicited user input defining at least a master/detail presentation pattern [29/11-18 and 31/12-18] for every class of the system specifying how users of said target computer program will be able to query the state of a given instance of a class of said target computer program owning said master/detail presentation pattern [29/11-18 and 31/12-18] (or the population of a class of said target computer program owning said master/detail presentation pattern), and the population of instances of classes related with an instance of the class of said target computer program owning said master/detail presentation pattern with said solicited user input further comprising user input which defines:

an instance presentation pattern [31/14] , or a class population presentation pattern [31/14-15] , defined for the class owning said master/detail presentation pattern, referred to as "master presentation pattern";

a set of presentation patterns which may be of type instance presentation pattern, class population presentation pattern, or master/detail presentation pattern [31/14-18], defined for classes related with the class owning the master/detail presentation pattern, referred to as "detail

presentation patterns"; and

converting said user input into one or more data structures [5/3] implementing said instance presentation pattern.

GROUNDS OF REJECTIONS TO BE REVIEWED ON APPEAL

The Examiner's *sua sponte* withdrawal of claims 26-73 from consideration is appealed as is the rejection under 35 U.S.C. Section 102 of claim 25.

ARGUMENT

The Improper Restriction Requirement

The Examiner has made a mistake in asserting that claims 26-73 are directed to separate and distinct inventions from claim 25 because claims 26-73 are directed to subcombinations within the scope of independent claim 25.

37 CFR 1.142(b) and MPEP Setion 821.03 were cited by the Examiner as authority for taking his position.

37 CFR 1.142(b) states:

> (b) Claims to the invention or inventions not elected, if not canceled, are nevertheless withdrawn from further consideration by the examiner by the election, subject however to reinstatement in the event the requirement for restriction is withdrawn or overruled.

MPEP 821.03 states:

> "Claims added by amendment following action by the examiner, MPEP § 818.01, § 818.02(a), to an invention other than previously claimed, should be treated as indicated by
> 37 CFR 1.145.
>
> 37 CFR 1.145 Subsequent presentation of claims for different invention.
>
> If, after an office action on an application, the applicant presents claims directed to an invention distinct from and independent of the invention previously claimed, the applicant will

> be required to restrict the claims to the invention previously claimed if the amendment is entered, subject to reconsideration and review as provided in §§ 1.143 and 1.144 ...."

What Is A Separate and Distinct Invention?

For a restriction requirement to be proper, the restricted claim classes must define independent and distinct inventions. MPEP section 802.01 defines what "independent and distinct" means as follows (bold emphasis added to sections that support our argument that the invention classes drawn by the Examiner are not in fact independent and distinct):

> 802.01 Meaning of "Independent" and "Distinct" [R-3]
>
> 35 U.S.C. 121 quoted in the preceding section states that the Director may require restriction if two or more "independent and distinct" inventions are claimed in one application. In 37 CFR 1.141, the statement is made that two or more "independent and distinct inventions" may not be claimed in one application.
>
> This raises the question of the inventions as between which the Director may require restriction. This, in turn, depends on the construction of the expression "independent and distinct" inventions.
>
> "Independent", of course, means not dependent. If "distinct" means the same thing, then its use in the statute and in the rule is redundant. If "distinct" means something different, then the question arises as to what the difference in meaning between these two words may be. The hearings before the committees of Congress considering the codification of the patent laws indicate that 35 U.S.C. 121: "enacts as law existing practice with respect to division, at the same time introducing a number of changes."
>
> The report on the hearings does not mention as a change that is introduced, the inventions between which the Director may properly require division.
>
> The term "independent" as already pointed out, means not dependent. **A large number of inventions between which, prior to the 1952 Act, division had been proper, are dependent inventions , such as, for example, combination and a subcombination thereof;** as process and apparatus used in the practice of the process; as

composition and the process in which the composition is used; as process and the product made by such process, etc. If section 121 of the 1952 Act were intended to direct the Director never to approve division between dependent inventions, the word "independent" would clearly have been used alone. If the Director has authority or discretion to restrict independent inventions only, then restriction would be improper as between dependent inventions, e.g., the examples used for purpose of illustration above. Such was clearly not the intent of Congress. Nothing in the language of the statute and nothing in the hearings of the committees indicate any intent to change the substantive law on this subject. On the contrary, joinder of the term "distinct" with the term "independent", indicates lack of such intent. The law has long been established that dependent inventions (frequently termed related inventions) such as used for illustration above may be properly divided if they are, in fact, "distinct" inventions, even though dependent.

I. INDEPENDENT

The term "independent" (i.e., **not dependent) means that there is no disclosed relationship between the two or more inventions claimed, that is, they are unconnected in design, operation, and effect**. For example, a process and an apparatus incapable of being used in practicing the process are independent inventions. See also MPEP § 806.06 and § 808.01.

II. DISTINCT

Two or more inventions are related (i.e., not independent) if they are disclosed as connected in at least one of design (e.g., structure or method of manufacture), operation (e.g., function or method of use), or effect. **Examples of related inventions include combination and part (subcombination)** thereof, process and apparatus for its practice, process and product made, etc. In this definition the term related is used as an alternative for dependent in referring to inventions other than independent inventions.

Related inventions **are distinct if the inventions as claimed are not connected in at least one of design, operation, or effect** (e.g., can be made by, or used in, a materially different process) and wherein at least one invention is PATENTABLE (novel and nonobvious) OVER THE OTHER (though they may each be unpatentable over the prior art). See MPEP § 806.05(c) (combination and subcombination) and § 806.05(j)

> (related products or related processes) for examples of when a two-way test is required for distinctness.
>
> It is further noted that the terms "independent" and "distinct" are used in decisions with varying meanings. All decisions should be read carefully to determine the meaning intended.

Based upon the above rules of law, it is clear that a restriction requirement is proper only when the inventions are independent and distinct. Inventions are not independent if they are disclosed as connected by at least one of design (structure), operation (function) or effect. An example of related inventions is combination and subcombination. If a claim is to a subcombination, such as a claim directed to one element of a combination claim which works in the combination to achieve the overall result of the combination, the subcombination claim is not independent from the combination claim because the part is related to the whole.

Distinct inventions are not connected by design (structure), operation or effect. A subcombination of a combination is one piece of a larger whole. As such, the subcombination is related by structure to the combination as the subcombination is part of the structure of the combination. The two are related by design. If the combination would not work to achieve its intended result without the subcombination or part, then the two inventions are neither distinct nor independent and they cannot properly be divided.

Claims 26 through 73, which the Examiner *sua sponte* withdrew from consideration, are each a subcombination of claim 25, and, as such, claims 26 - 73 are not distinct from the invention of claim 25 because each is a species of a particular part of the combination of claim 25. The following paragraphs explain the specifics of this argument as to each of claims 26 through 73.

Claim 26

Claim 26 depends from claim 25 and calls for the instructions recorded on the medium to control the computer to receive user input which defines four different

types of presentation patterns. These patterns are part of the presentation model which claim 25 requires as part of the conceptual model, so claim 26 is structurally part of claim 25 and not distinct.

Claim 27

Claim 27 depends from claim 26 and calls for the instructions to control the computer to receive presentation pattern for action selection which defines heirarchically how users access said service, instance and class population and master-detail presentation patterns. These patterns are part of the presentation model which claim 25 requires as part of the conceptual model, so claim 27 is structurally part of claim 25 and not distinct.

Claim 28

Claim 28 depends from claim 27 (which depends from claim 26 which depends from claim 25) and calls for instructions to control the computer to receive user data to define said service, instance, class population and master-detail presentation patterns. These patterns are part of the presentation model which claim 25 requires as part of the conceptual model. Claim 28 defines a part or subcombination of the medium and instructions defined by claim 25 so the two are structurally related and not distinct.

Claim 29

Claim 29 depends from claim 25 and calls for instructions that control a computer to receive user input data which defines classes within the object model of claim 25 which include services (which are either events or transactions) which include at least a creation event to create an instance of a class and link it to other related classes within the object model. These instructions are part of the instructions called for by claim 25 to control the computer to create an object model, so claim 29 is a subcombination or part of the instruction set or structure defined by claim 25 and is not distinct.

Claim 30

Claim 30 depends from claim 25 and calls for instructions which control the computer to receive user input data which defines the functionality of transaction type services. Services are part of every class which is part of the object model which is built by the user under control of the instruction set defined by claim 25 so these claim 30 instructions are part or a subcombination of the instructions defined by claim 25 and are not distinct.

Claim 31

Claim 31 depends from claim 25 and calls for instructions which control the computer to receive user input data which defines preconditions, integrity constraints, valid object lives, agent relationships and trigger relationships all of which are part of the valuation formulas of events which effect the value of attributes of classes and formulas of transactions which are services of classes, classes being part of the object model created by the user under control of the instructions of claim 25. The instructions of claim 31 are therefore a subcombination of the instructions of claim 25 and are not distinct.

Claim 32

Claim 32 depends from claim 31 and defines instructions which control a computer to receive user input which defines preconditions as conditions which must be satisfied for a service of a class which is part of the object model to execute. These instructions are a subcombination of the instructions of claim 25.

Claim 33

Claim 33 depends from claim 31 and calls for instructions which control a computer to receive user input which defines integrity constraints which are part of the object model of claim 25 so that instances of objects in a class cannot have their states changed to non allowed states. These instructions are a subcombination of the instructions of claim 25.

Claims 34-36

Claims 34 - 36 all depend from claim 31 and define instructions which control

a computer to receive user input that: defines valid lives for instances of classes within the object model of claim 25; defines agent relationships between agent and server classes within the object model; defines trigger relationships which specify mandatory execution of a service within a class of the object model of claim 25. All these instruction sets are subcombinations of the instruction set of claim 25 as each defines a part of the object model of claim 25.

Independent Claim 37

Independent claim 37 calls for a computer-readable medium having stored thereon a data structure which is comprised of a collection of fields of data that define the conceptual model (same conceptual model as claim 25). The conceptual model is comprised of an object model, dynamic model, functional model and presentation model. These are the same models as are created by the user data entered under control of the instructions stored on the medium of claim 25. As such claim 37 is not distinct from claim 25 in that it is related by process and product made. In other words, the instructions stored on the medium of claim 25 control a computer to carry out a process. That process creates the data structure of claim 37. As such, per the example of MPEP 802.1 that process and product made are not distinct inventions, restriction of claim 37 from claim 25 is not proper.

Claim 38

Claim 38 depends from claim 37 and defines the data structures to have user data which defines three types of events for each class which is defined in the object model of claim 37: creation, destruction and modification. As such, these data structures are part of the data structure of claim 37 as subcombination and combination of which it is a part.

Claim 39

Claim 39 is an independent claim which calls for a computer readable medium storing instructions which control a computer to carry out a process to receive user input to define data structures which define the primitives of a conceptual model

comprised of an object model, dynamic model, functional model and presentation model, automatically convert the data structures to a formal language specification that defines a complete software program which is the functional equivalent of the conceptual model. These two process steps are the same as claim 25 so claim 39 is related to claim 25 in that claim 25 is a species of the genus of claim 39 (which does not include the validation step). A species can hardly be said to be independent and distinct from its genus.

Claim 39 includes other limitations which define the functionality of the conceptual model in terms of services taking the form of local or global transactions which are defined functionally by formulas which express which services the transaction encompasses and the value assigned to every argument of every service. Claim 39 also includes limitations which require instructions to control the computer to receive user input which defines valuations for the functional model which define the effect events of a class have on the value of a variable attribute of the class.

Claim 40

Claim 40 depends from claim 39 and calls for instructions which control a computer to solicit user input defining different valuations for each variable attribute in the functional model of claim 39. The functional model is part of the conceptual model recited in claim 39 and claim 25, so claim 40 defines a set of instructions on the medium which is a subcombination of the set of instructions defined in claims 25 and 39.

Claims 41 - 43

Claims 41-43 each depend directly from 39, and define instruction sets which: control a computer to solicit user input which define valuation data structures that define valuations of different categories (push-pop, state independent etc.) for each variable attribute of an object in a class; such that each variable attribute can include a list of valuation data structures that define how the variable attribute's value and the object's state is changed by different events; such that each valuation data structure

defines an optional condition that must be met before the valuation will be allowed to affect the value of the variable attribute.

All these instruction sets are subcombinations of the instruction set of claims 25 and 39 because the instruction sets of claims 25 and 39 control a computer to solicit user input to define data structures which define an object model with classes with variable attributes and a functional model which defines how valuation formulas affect the values of variable attributes when events occur. The instruction sets of claim 41 and 43 define species of instruction sets which solicit different types of user input data to define different species of functional model data structures within the class of species defined by claims 25 and 39. Thus, the inventions of claim 41-43 are not distinct from the inventions of claims 25 and 39.

Claims 44 - 46

Independent claim 44 calls for computer instructions stored on a computer-readable medium which control the computer to receive user input which defines a creation event (a creation event is needed in every class in the object model of the Conceptual Model) and optional destruction and modification events (also part of the object model when present - modification events have valuations). The claim further defines what the user input defines about the creation event and how it creates an instance of a class owning the creation event and defines values for all constant and variable attributes. The claim defines in general what process steps are carried out by the instructions which are stored on the medium and how they control the computer to receive the user input which defines various things about a class within the object model within the conceptual model. As such, these instructions and process steps are a subsystem or species of the process steps and instructions of claims 25 which defines an instruction set which controls a computer to receive user input data which defines a Conceptual Model which includes an object model which includes the classes created by the subsystem claim 44. The inventions of claims 25 and 44 are not distinct.

Claims 45 and 46 further define the instruction set and process steps of claim 44 as soliciting user input which: defines the valuations as push-pop, state-independent or discrete domain; and which is entered via suitable user interface mechanisms. These claims just add more detail to the subcombination claim 44 and are not distinct inventions from the invention of claim 25 for the same reason independent claim 44 is not distinct.

Claims 47 - 56

Independent claim 47 and its dependent claims 48 - 56 do not define an invention that is distinct from the invention of claim 25 because these claims define subcombination species of mediums with instructions sets which control the computer to receive user input defining valuations which are part of the functional model which is part of the overall system defined by claim 25. Specifically, these valuations are included within and are a subcombination of the system defined by the broad definition of the functional model as a system which allows a SOSY modeler (a person) to specify a class, an attribute of that class and an event of that class and then define a mathematical or logical formula that defines how the attribute's value will be changed when this event happens. The functional model is part of the system defined by claim 25 so the subsystem claims 47 -56 that define various species of instruction sets and processing caused by those instruction sets to define valuations are not distinct inventions. Claims 48 - 56 define various aspects of the valuations created by the user input solicited in the process controlled by the instructions on the medium.

Claims 57 - 63

Independent claim 57 and its dependent claims 58 - 63 do not define an invention that is distinct from the invention of claim 25 because these claims define subcombination species of instruction sets which control the computer to solicit user input that define transactions which are part of the object model which is part of the overall system defined in claim 25. Specifically, transactions are groupings of

services in execution units that comprise processes which change the state of objects in the object model defined in claim 25 [specification 30/28 -32]. Transactions are expressed as formulas that specify sequences or alternations of services. That transactions are part of the services of classes of objects in the object model is proven by the following quotation from the specification at 39/29 - 32:

> The system classes are obtained from the object model. For each class, there are a set of constant, variable or derived attributes; a set of services, including private and shared events and local transactions; integrity constraints specified for the class; and derivation expressions corresponding to the derived attributes.

Therefore claim 57 defines a subcombination of the system defined in claim 25 and is not distinct. Dependent claim 58 - 63 only define species within the subcombination defined by claim 57. For example, claim 58 defines instructions stored on the medium which control the computer to solicit user input that defines transactions in terms of a sequence or alternation of services, and claim 59 defines the transactions in terms of the "all or nothing" policy.

Claims 64 - 66

Independent claim 64 is not a distinct invention from claim 25 because it defines a subcombination of instructions stored on a medium which control a computer to solicit user input which defines the primitives of the presentation model which is one of the four models defined in claim 25 which together comprise the Conceptual Model. As such, the relationships of claim 64 to claim 25 is subcombination to combination. Dependent claims 65 and 66 define species of the subcombination in defining the different types of

patterns that are defined by the user input solicited by the instructions on the medium.

Claims 67 - 73

Independent claim 67 defines a subcombination of the system defined in claim 25 because it calls for computer instructions stored on the medium which solicit user input to define primitives of a presentation model which is part of the Conceptual Model defined in claim 25. The user input defines the presentation model in terms of a set of patterns that define how users will be able to interact with the final computer program defined by the Conceptual Model. Claim 67 is therefore related to claim 25 as subcombination and combination or as a part is to the whole which contains that part. Claims 68 and 69 depend from claim 67 and further define the types of patterns the instructions control the computer to solicit user input data for. In other words, the presentation model of claim 25 defines patterns. Each pattern requires user input to define things like to class is the pattern to be applied, which services of a class to present, etc.

Independent claims 70, 71 and 72 also define a subcombination of the system defined in claim 25 because each of these claims calls for instructions stored on the medium which control the computer to solicit user input which defines the presentation model part of the Conceptual Model of claim 25. Claim 70 defines specific types of optional patterns, while claim 71 calls for soliciting user input which defines at least an instance presentation pattern for every class defined in the object model. Claim 72 calls for soliciting user input which defines a class population presentation pattern for every class in the object model. All these are different species

of the instructions of independent claim 67 and all are subcombinations or parts of the system defined by claim 25 and are not distinct inventions.

The Section 102 Rejection Of Claim 25

Fore a rejection under 35 USC 102 to be proper, every limitation of the claim must be found in a single prior art reference. Here, the Examiner Ryu et al, U.S. 5,481,718 as anticipating claim 25.

Claim 25 reads:

25. [NEW] A computer-readable medium having stored thereon computer-readable instructions, which, when executed by a computer, cause said computer to perform the following process:

receive user input data that defines primitives of a conceptual model, said conceptual model being comprised of an object model, a dynamic model, a functional model and a presentation model which together define the complete functionality of a target computer program to be automatically generated and which defines complete interface mechanisms for interaction of user or other processes with the functionality of said target computer program, no other software code, code component, code libraries or any other third party software artifact being necessary to completely define the functionality of said target computer program and its interface mechanisms;

automatically converting said user input data into data structures in the form of formal language statements organized by a formal language syntax, the collection of all such formal language statements forming a formal language specification;

validating said formal language specification to ensure it is complete, correct and unambiguous to prepare said formal language specification for automatic translation into complete, operative code of said target computer program which is the functional equivalent of said conceptual model and having a user interface defined by the primitives entered by said user which define said presentation model.

Claim 25 is a computer-readable medium that stores computer-readable instructions which control a computer to implement a computer program modeler which can be used to design a computer program to be built by automatic translation of a formal language specification. The formal language specification is automatically created from a Conceptual Model which is defined by user input data solicited by the process carried out by the computer under control of the instructions on the medium. The user input data is solicited via user interface mechanisms such as dialog boxes, etc. The user input data defines a Conceptual Model of the program to be automatically written. That Conceptual Model is comprised of an object model, a dynamic model, a functional model and a presentation model. The object model defines the classes of objects in the system. The dynamic model defines service preconditions via a state transition diagram, the service preconditions being formulas labeling the state transitions and defines valid object lives from creation to death. The dynamic model also defines an object interaction diagram which defines trigger relationships and global transactions which are services implemented by an object in the object model which affect other objects in the object model. The functional model contains formulas entered by the user which define how events affect the values of variable attributes in objects defined in the object model. The user input which defines the presentation model defines the types of patterns of graphical user interface tools and other user interface mechanisms which will constitute the user interface of the program under design.

Ryu only teaches an object-oriented processing system which:

1) overcomes the problem in the prior art object-oriented language which are generally data oriented in dealing with resources by providing a system which is capable of treating methods as an object in addition to data;

2) overcomes the problem of the prior art providing no design principle with regard to the attempt to handle new processing with a new objective or target by carrying out designing by using the static model, dynamic model and the functional

model;

3) overcomes the problem of the prior art of providing no substantial support to the attempt of systematically considering the causality by only providing instructions which can handle causality within a class by providing a system which can handle causality both within a class and outside the class.

4) overcome the limitations of the prior art which can only define structure in the form of "is-a" relationships or "part-of" relationships by providing a system which is capable of using relation links and network links.

5) overcomes the problem in the prior art of using pointers to set links which slows down the system by providing a system which uses a high speed internal schema.

6) overcomes the problem in the prior art of not supporting composite objects by providing a system which supports composite objects.

7) overcomes the limitation of the prior art methods which define how to use individual information which are limited to predetermined content by providing a system which provide autonomy to the methods such that they can inform the operator of various conditions that the operator should know.

Ryu does not teach soliciting user input which defines the primitives of a presentation model which defines the desired user interface of the program under design as called for by claim 25. Claim 25 calls for:

> receive user input data that defines primitives of a conceptual model, said conceptual model being comprised of ...a presentation model which together define the complete functionality of a target computer program to be automatically generated and which defines complete interface mechanisms for interaction of user or other processes with the functionality of said target computer program....;

Since this claim element is not taught by Ryu because there is no mention of a presentation model, claim 25 is not anticipated.

Ryu does not teach converting his models into a formal language specification. Claim 25 calls for:

automatically converting said user input data into data structures in the form of formal language statements organized by a formal language syntax, the collection of all such formal language statements forming a formal language specification;

Claim 25 is not anticipated for this reason.

Ryu does not teach a Dynamic Model or a Functional Model that are the same data structures as the Dynamic Model and Functional Modle of claim 25.

Ryu's Dynamic Model indicates the time sequential relationship (order of processing) between instances forming the classes as a session. The Ryu static model describes a set of classes and relationships between them, whose instantiation in terms of instances of classes forms the dynamic model. In other words, the Ryu Dynamic Model is the "run-time" version of the static model, and the terms "static" and "dynamic" are used in this sense.

In contrast, the Dynamic Model of claim 25 defines service preconditions and state transitions and defines the valid lives of objects, trigger relationships and global transactions. This is not the same thing as Ryu's Dynamic Model because the concepts of a state transition diagram that defines event preconditions and event transitions and state transitions and triggers and global transactions are not taught in Ryu.

Likewise, Ryu's Functional Model is a set of classes and methods related to existing behavior.

In contrast, the Functional Model of claim 25 defines the valuation formulas that define the effect of events on the values of variable attributes. Ryu does not teach the concept of valuation formulas, so the Functional Model of claim 25 is radically different in structure and operation than the Functional Model in the Ryu system.

In Col. 11, lines 8-18, Fyu teaches the use of semantic data related to the nature of the object, but it does not teach the semantics of a formal language specification which is what the user data in the system of claim 25 is turned into. **In other words, Ryu does not teach a medium storing instructions which control a**

computer to solicit and receive user input data which is automatically converted into semantic data of a formal language specification as called for by claim 25.

Another difference between the system of claim 25 and the Ryu prior art is in validation. The system of claim 25 converts user input which defines the primitives of a Conceptual Model into a formal language specification which has rules of syntax and semantics. These rules are used in the system of claim 25 to validate the formal language specification to ensure it is complete and correct prior to automated translation into working code.

In contrast, Ryu does not teach the use of a formal language specification nor any other type of formalism and there is no mention in Ryu anywhere of rules of syntax or semantics of any specification as being part of his system nor of use of rules of syntax and semantics to validate a specification to ensure it is complete and correct. Therefore, claim 25 is not anticipated because Ryu does not teach any validation step.

The Examiner stated the ability of the system of claim 25 of "automatically converting said user input into data structures in the form of formal language statements organized by a formal language syntax, the collection of all such formal language statements forming a formal language specification" is anticipated by Col. 9, lines 49-53. That passage of Ryu refers to how Ryu's dynamic model is formed by means of a causality relationship with the static model. **This passage of Ryu does not teach automatic conversion of user input data into a formal language specification as required by claim 25.**

Further, Ryu does not teach any kind of conversion, transformation or translation.

The Examiner asserted that the validation process required by claim 25 is anticipated by the "simulation" taught in Col. 10, Lines 42-51. However, validation and simulation do not mean the same thing and are not the same processes. Validation is the process of checking to see if something satisfies a certain criterion, while

"simulation" is an imitation of some real device or state of affairs. Simulation attempts to represent certain features of the behavior of a physical or abstract system by the behavior of another system. **Ryu's simulation is not a validation process.**

There is no evidence that the Col. 11, lines 42-52 teachings link the "hyper language processing unit" with validation tasks so the "hyper language processing unit" does not perform the validation process called for by claim 25.

Finally, the test process taught by Ryu in Col. 10, line 62 to Col. 11, line 2 is not a validation process because the testing acitivity is related to "executionable process data" and not to "formal language specifications".

Claim 25 is therefore not anticipated.

CLAIMS APPENDIX

25. [NEW] A computer-readable medium having stored thereon computer-readable instructions, which, when executed by a computer, cause said computer to perform the following process:

receive user input data that defines primitives of a conceptual model, said conceptual model being comprised of an object model, a dynamic model, a functional model and a presentation model which together define the complete functionality of a target computer program to be automatically generated and which defines complete interface mechanisms for interaction of user or other processes with the functionality of said target computer program, no other software code, code component, code libraries or any other third party software artifact being necessary to completely define the functionality of said target computer program and its interface mechanisms;

automatically converting said user input data into data structures in the form of formal language statements organized by a formal language syntax, the collection of all such formal language statements forming a formal language specification;

validating said formal language specification to ensure it is complete, correct and unambiguous to prepare said formal language specification for automatic translation into complete, operative code of said target computer program which is the functional equivalent of said conceptual model and having a user interface defined by the primitives entered by said user which define said presentation model.

26. [NEW] The computer-readable medium of claim 25 wherein said computer
readable instructions control said computer to receive user input which defines one or more service presentation patterns, one or more instance presentation

patterns, one or more class population presentation patterns and one or more master-detail presentation patterns as part of said interface mechanisms

27. [NEW] The computer-readable medium of claim 26 wherein said computer-readable instructions control said computer to receive user input which defines an action selection presentation pattern which defines hierarchically how users access said one or more service presentation patterns, one or more instance presentation patterns, one or more class population presentation patterns and one or more master-detail presentation patterns.

28. [NEW] The computer-readable medium of claim 27 wherein said computer-readable instructions control said computer to receive user input which defines said service presentation patterns, said instance presentation patterns, said class population presentation patterns and said master-detail presentation patterns by means of definition of other auxiliary primitives including but not limited to introduction patterns, defined selection patterns, dependency patterns, population selection patterns, display set patterns, filter patterns and order criterion patterns.

29. [NEW] The computer-readable medium of claim 25 wherein said computer-readable
instructions control said computer to receive user input data which defines classes within said object model each of which has services which may be events or transactions and include at least a creation event to create an instance of said class and create links of said instance to instances of classes related to said class, an optional destruction event which breaks links of a given instance of said class to instances of related classes and destroys said given instance of said class, and an optional set of modification events whose functionality is defined in

said functional model by means of valuations where a valuation is a primitive which creates a relationship between a variable attribute and an event of a class so as to define how the occurrence of said event affects the value of said variable attribute.

30. [NEW] The computer-readable medium of claim 25 wherein said computer-readable instructions control said computer to receive user input data which defines the functionality of transaction type services, which can be local or global, in terms of a sequence and/or alternation of other services, said services being events and/or transactions, with the functionality of transactions defined by a formula which defines which services the transaction encompasses and the value assigned to every argument of every service comprising said transaction.

31. [NEW] The computer-readable medium of claim 25 wherein said computer-readable instructions control said computer to receive user input data which defines secondary primitives which enrich the functionality of valuations of events and formulae of both local and global transactions by constraining said functionality using constraint primitives which include but are not limited to preconditions, integrity constraints, valid object lives, agent relationships and trigger relationships.

32. [NEW] The computer-readable medium of claim 31 wherein said computer-readable instructions control said computer to receive user input data which defines said preconditions as a condition which must be satisfied for a service to which said precondition is associated to execute, when executed by a given agent class, along with an error message to be displayed when said condition is not satisfied.

33. [NEW] The computer-readable medium of claim 31 wherein said computer-readable instructions control said computer to receive user input data which defines said integrity
constraint as a condition which must be satisfied at any given instant by instances of a class so as to prevent services from changing the state of objects in a class to a non-desired state, along with an error message to be displayed when said condition is not satisfied.

34. [NEW] The computer-readable medium of claim 31 wherein said computer-readable instructions control said computer to receive user input data which defines valid lives for instances of a class by defining if a service specified by said user input can be executed by a given agent on a given instance of said class depending on the services that were previously executed on said given instance so as to constrain the execution of services on instances based on the history of services previously executed on said instances.

35. [NEW] The computer-readable medium of claim 31 wherein said computer-readable instructions control said computer to receive user input data which defines a relationship between two classes one playing the role of the "agent class" and the other playing the role of the "server class" so as to designate which attributes of said server class will be observable by said agent class and which services of said server class will be executable by said agent class so as to enable a user to enter input data which limits the accessibility of users who log onto the system as instances of said agent class to functionality of said target computer program such that some services can be blocked from access by some users and which limits accessibility to the state of the system such that some attributes of a class or entire classes can be blocked from query by designated users.

36. [NEW] The computer-readable medium of claim 31 wherein said computer-readable instructions control said computer to receive user input data which defines trigger relationships to enrich the functionality of the system by specifying mandatory, and unnoticeable to the user, execution of a service whenever a Boolean condition holds, said condition specified on the state of an object.

37. [NEW] A computer-readable medium having stored thereon a data structure comprising:

a collection of fields that store data entered by a user defining a conceptual model of a computer program that the user wants automatically written by a translator process running on a computer, said conceptual model comprised of:

an object model data structure which defines a system class architecture comprised of one or more classes of objects defined by user input, each class having attributes the value of which collectively define the state of a system defined by said conceptual model (hereafter the target program);

a dynamic model data structure which defines valid object life cycles and which interobject communications can be established and defined by user input;

a functional model data structure which defines the semantics associated with any change of an object state as a consequence of an event occurrence, user input defining data structures which define the class and which variable attribute of said class is affected by which event of that class and how the event affects the value of said variable attribute, said user input also defining a mathematical or

logical valuation that defines how said variable attribute's value will be changed when said event happens;

a presentation model data structure defined by user input and which defines a full user interface defining how users or other processes will be able to interact with the functionality of the target program, said user input defining a set of patterns that specify which services will be available to user of the system and which information about the state of the target program users of the target program will be able to query.

38. The computer-readable medium of claim 37 wherein said data structures have user input data which define three types of events:

a creation event for each class which creates an instance of the class owning said creation event and provides values to all constant and variable attributes of the class which are required upon creation and establishing all relationships with instances of classes which are required to have a relationship with the instance of the class owning said creation event;

a destruction event for each class which eliminates all relationships between a given instance of the class owning said destruction event and other classes having relationships with the class owning said destruction event; or

a modification event for each class which modifies the values of one or more variable attributes of a given instance of the class owning said modification event, the effect of said modification being defined by one or more valuations linked to said one or more variable attributes of said given instance of said class owning said modification event.

39. [NEW] A computer-readable medium having stored thereon computer-

readable instructions which, when executed by a computer cause said computer to carry out the following process:

soliciting and receiving user input and converting said user input to data structures which define primitives of a conceptual model comprising an object model comprising one or more classes of objects which have attributes the state of which collective define the state of the system, and a dynamic model, and a functional model and a presentation model;

automatically converting said data structures of said conceptual model into statements according to the syntax and semantics of a formal language, said collection of formal language statements defining a formal language specification which defines the functionality of a computer program to be automatically written (hereafter the target program) by a computer executing a translator process, said target program being a full computer program having a user interface specified by data structures comprising said presentation model, and not a prototype which needs additional code, code components, code libraries or any third party software artifact and which will be the full functional equivalent of said conceptual model;

and wherein said data structures define the functionality of said conceptual model in terms of services which change the state of a system implemented by said target program, said services being events or local or global transactions, and wherein events are defined as the smallest execution units possible in the scope of a class within said object model in that it is not possible to decompose an event into more elementary execution units, and wherein said local or global transactions are molecular execution units that have their functionality defined in terms of a sequence and/or alternation of other services each of which can be either an event or transaction, and wherein the functionality of a local or global transaction is

explicitly defined by a data structure which defines a formula which expresses which services the transaction encompasses and the value assigned to every argument of every service comprising the transaction, and wherein said data structures which define local transactions limit the effect of a local transaction to affecting the state of primarily an instance of the class owning said transaction and potentially instances of classes related with the class owning said transaction, and wherein global transactions are not limited to affecting the state of only one instance of a class and can affect the state of any instance or set of instances of any class or set of classes;

and wherein said solicited user input includes user input which defines valuations for said functional model, where each valuation is a primitive relating a variable attribute and an event of a class and defines how the occurrence of said event affects the value of said variable attribute thereby explicitly defining the functionality of said event.

40. [NEW] The computer-readable medium of claim 39 wherein said computer-readable instructions, when executed by a computer, control said computer such that solicitation of user input defining said valuations is such that user input defining data structures which define different valuations for each variable attribute can be entered, each valuation relating said each variable attribute with the same or different events.

41. [NEW] The computer-readable medium of claim 39 wherein said computer-readable instructions, when executed by a computer, control said computer to solicit user input defining valuation data structures which can define valuations of different categories for each said variable attribute.

42. [NEW] The computer-readable medium of claim 39 wherein said computer-readable instructions, when executed by a computer, control said computer to solicit user input defining valuation data structures such that each variable attribute can include a list of valuation data structures that define how the variable attribute's value, and therefore the object's state, is changed by means of the different events.

43. [NEW] The computer-readable medium of claim 39 wherein said computer-readable instructions, when executed by a computer, control said computer to solicit user input defining valuation data structures such that each valuation data structure defines an optional condition that must be satisfied to apply the effect of the valuation on the value of the attribute, the event that will cause the valuation to be executed and the effect of the event to a specified attribute.

EVENT CLAIMS

44. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer cause said computer to perform the following process:

receive user input which defines a creation event an optional destruction event and an optional set of modification events for a class which is to become part of an object model of a conceptual model of a computer program to be automatically written, said user input defining said creation event creating an instance of a class owning said creation event and providing a value for all constant and variable attributes of the class which are required upon creation, said user input also establishing all required relationships with instances of classes related with the instance of the class owning said creation event, and wherein said user input defines a

destruction event which eliminates all relationships between a given instance of a class owning said destruction event and related instances of classes which have relationships with said class and then destroying the instance of the class owning said destruction event, and wherein said user input defines a set of modification events each of which uses user input to modify the values of one or more variable attributes of a given instance of the class owning said modification event, the effect of said modification defined by one or more logical or mathematical valuations defined by user input and which affect the value of one or more variable attributes upon the occurrence of said modification event;

converting said user input into data structures which define an object model having a class having said creation, destruction and modification events.

45. [NEW] The computer-readable medium of claim 44 wherein said computer-readable instructions stored thereon are such that they cause a computer executing them to solicit user input which defines said valuations as push-pop, state-independent or discrete domain.

46. [NEW] The computer-readable medium of claim 44 wherein said computer-readable instructions stored thereon are such that they cause a computer executing them to solicit user input which defines said creation, destruction and modification events and said valuations, said solicitation by means of any suitable user interface mechanisms which prompt a user for the needed input to define said creation, destruction and modification events and said valuations.

VALUATION CLAIMS

47. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer cause said computer to perform the following process:

receiving user input to define a valuation which will affect the value of a variable attribute of a class upon the occurrence of an event and the satisfaction of an optional condition, said solicited user input also defining which variable attribute of which class will be affected and the event which will trigger said valuation to affect the value of said variable attribute; and

converting said user input into data structures which define said valuation as part of a conceptual model of a computer program to be automatically written.

48. [NEW] The computer-readable medium of claim 47 wherein said computer-readable instructions stored thereon are such as to control said computer to solicit user input to define the effect of said valuation with a formula referred to as a "valuation effect" using constants, values of input arguments of said event, values of other attributes of the class owning the variable attribute or an ancestor of it, or values of attributes of classes or ancestors of classes related to the class owning the variable attribute, and to convert said solicited user input into one or more data structures which define said valuation effect formula.

49. [NEW] The computer-readable medium of claim 47 wherein said computer-readable instructions stored thereon are such as to control said computer to solicit user input to define said optional condition for the valuation with a Boolean formula referred to as a "valuation condition" that may be formed using constants, values of input arguments of said event, values of other attributes of the class owning the variable attribute or an ancestor of it, or values of attributes of classes or ancestors of classes related with the class owning the variable

attribute, and converting said user input into data structures which define said Boolean valuation condition formula.

50. [NEW] The computer-readable medium of claim 47 wherein said computer-readable instructions stored thereon are such as to control said computer to solicit user input to define said valuation such that it has both a valuation effect formula and a valuation condition formula, said solicited user input defining said valuation effect formula and said valuation condition formula by defining the valuation effect formula the value said variable attribute affected by said valuation will take upon occurrence of said event providing the valuation condition formula evaluates to a Boolean value of true, and to control said computer to use said solicited user input to form one or more data structures which define said valuation which has both said valuation effect and said valuation condition.

51. [NEW] The computer-readable medium of claim 47 wherein said computer-readable instructions stored thereon are such as to control said computer to solicit user input to define said valuation such that it has only a valuation effect formula, and no valuation condition formula defining, by means of said valuation effect formula, the value said variable attribute will take upon occurrence of said event unconditionally, and controlling said computer to convert said solicited user input into one or more data structures which implement said valuation in a conceptual model.

52. [NEW] The computer-readable medium of claim 47 wherein said computer-readable instructions stored thereon are such as to control said computer to solicit user input to define said valuation such that it falls within one of the following three categories: push-pop, state-independent and discrete-domain,

and for controlling said computer to convert said user input into one or more data structures which implement said valuation in the selected category.

53. [NEW] The computer-readable medium of claim 47 wherein said computer-readable instructions stored thereon are such as to control said computer to solicit user input to define said valuation such that it falls within a push-pop category, said push-pop valuation being a valuation where said event which triggers said valuation increases or decreases the value of the variable attribute affected by said valuation by a given quantity or resets the value of the variable attribute affected by said valuation to a certain value, and controlling said computer to convert said solicited user input into one or more data structures which implements said push-pop valuation.

54. [NEW] The computer-readable medium of claim 47 wherein said computer-readable instructions stored thereon are such as to control said computer to solicit user input to define said valuation such that it falls within a state-independent valuation category, wherein said state-independent valuation is one whose valuation effect formula provides a new value to the variable attribute affected by said valuation regardless of the value said variable attribute has at the time the event which triggers said valuation occurs, and said computer-readable instructions controlling said computer to convert said solicited user input to create one or more data structures which implement said state-independent valuation.

55. [NEW] The computer-readable medium of claim 47 wherein said computer-readable instructions stored thereon are such as to control said computer to solicit user input to define said valuation such that it falls within a discrete domain category, wherein said discrete domain valuation is one whose valuation effect formula provides a new value for the variable attribute affected by

said valuation depending upon the current value of said variable attribute, and said computer-readable instructions controlling said computer to convert said solicited user input into one or more data structures which implement said discrete domain valuation.

56. [NEW] The computer-readable medium of claim 47 wherein said computer-readable instructions stored thereon are such as to control said computer to solicit user input to define one or more said valuations each of which affects the value of the same variable attribute upon occurrence of the same event, regardless of the category of said valuation, if and only if one of said valuations and only one of them has only a valuation effect formula but no valuation condition formula, and the rest of them compulsorily have both a valuation effect formula and a valuation condition formula, said computer-readable instructions for converting said solicited user input data into one or more data structures which implement said one or more valuations.

Transaction Claims

57. [NEW] A computer-readable medium having computer-readable instructions stored thereon, which, when executed by a computer, control said computer to solicit user input to define a transaction wherein said transaction is a molecular execution unit expressed in terms of a formula that specifies services in the form of events or transactions which together comprise said molecular execution unit, and said computer-readable instructions controlling said computer to convert said user input into one or more data structures which implement said transaction.

58. [NEW] The computer-readable medium of claim 57 wherein said

computer-readable instructions control said computer to solicit user input which defines said transaction in terms of a sequence and/or alternation of services, and said computer-readable instructions controlling said computer to convert said user input into one or more data structures which implement said transaction.

59. [NEW] The computer-readable medium of claim 57 wherein said computer-readable instructions control said computer to solicit user input which defines said transaction in terms of an "all-or-nothing" execution policy in the sense that the execution of every service comprising said transaction must be successful in creating changes to the state of the system for the execution of said transaction to be successful , and the failure in execution of any service comprising said transaction in creating changes to the state of the system means failure in the execution of said transaction and causing to override changes to be made to the state of the system in that the changes made to the state of the system by any of the services comprising said transaction will be reversed, said computer-readable instructions controlling said computer to convert said user input into one or more data structures implementing said "all-or-nothing" transaction.

60. [NEW] The computer-readable medium of claim 57 wherein said computer-readable instructions control said computer to solicit user input which defines said transaction as a local transaction wherein said one or more services which comprise said local transaction are events or transactions owned by the class or the ancestor of the class which owns said transaction which owns said transaction formula, or said one or more services which comprise said transaction are owned by classes related with the class owning the transaction which owns said transaction formula, and wherein said computer-readable instructions controlling said computer to convert said user input into one or more data

structures which implement said local transaction.

61. [NEW] The computer-readable medium of claim 57 wherein said computer-readable instructions control said computer to solicit user input which defines said transaction as a global transaction wherein said one or more services which comprise said global transaction are events and/or local transactions owned by any of the class in a conceptual model and/or global transactions.

62. [NEW] The computer-readable medium of claim 57 wherein said computer-readable instructions control said computer to solicit user input which defines said transaction as a local transaction, wherein said one or more services which comprise said local transaction have arguments whose value is determined in said transaction formula by contained formulas formed by constants, values of input arguments of said transaction, values of attributes of the class owning said transaction or an ancestor of it, or values of attributes of classes or ancestors of classes related with the class owning said transaction, and said computer-readable instructions controlling said computer to convert said user input into one or more data structures which implement said local transaction.

63. [NEW] The computer-readable medium of claim 57 wherein said computer-readable instructions control said computer to solicit user input which defines said transaction as a global transaction, wherein said one or more services which comprise said global transaction have arguments whose value is determined in said transaction formula by contained formulas formed by constants, values of input arguments of said transaction, or values of attributes of any class in said conceptual model, said computer-readable instructions also being such as to control said computer to convert said user input into one or more

data structures which implement said global transaction.

CLAIMS ABOUT SPECIFYING INTERFACE MECHANISMS TO INTERFACE TO
FUNCTIONALITY OF THE SYSTEM

64. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer control said computer to perform the following process:

soliciting user input to define primitives of a conceptual model defining the functionality of a target computer program to be written automatically, said conceptual model comprised of an object model, a dynamic model, a functional model and a presentation model;

converting said user input into data structures which implement said conceptual model in the form of statements in a formal language syntax that together comprise a formal language specification, said formal language specification defining a complete computer program having a user interface and which needs no additional code, code components, code libraries or any other third party software artifact in order to be functionally equivalent to the conceptual model;

and wherein said data structures and formal language specification define the functionality of the conceptual model in terms of how the state of the system is changed by means of services, said services being events and transactions where transactions are comprised of other services and falling in either a local or global category;

and wherein said data structures and formal language specifications define said presentation model in the form of user interface mechanisms which define how a user or other processes will be able to interact with the functionality of the

system.

65. [NEW] The computer-readable medium of claim 64 wherein said computer-readable instructions stored thereon are such as to solicit user input which defines said presentation model by specifying a set of patterns that specify the interaction means through which services will be available to user of the system and through which information about the state of the system users of the system will be able to query.

66. [NEW] The computer readable medium of claim 64 wherein said computer-readable instructions stored thereon are such as to solicit user input which defines said presentation model in terms of a set of service presentation patterns, instance presentation patterns, class population presentation patterns, master/detail presentation patterns and an action selection presentation pattern.

67. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer control said computer to perform the following process:

soliciting user input to define primitives of a presentation model which is part of a conceptual model of a target computer program to be automatically written, said presentation model defining a user interface by a set of patterns which will be used to define how users can interact with said target computer program by specifying the interaction scenarios for services which will be available to users of the target computer program and the interaction scenarios to query information about the state of the target computer program which users of the system will be able to query;

and converting said user input to one or more data structures which define said patterns of said presentation model.

68. [NEW] The computer readable medium of claim 67 wherein said computer-readable instructions stored thereon are such as to solicit user input which defines said presentation model in terms of a set of service presentation patterns, instance presentation patterns, class population presentation patterns, master/detail presentation patterns and an action selection presentation pattern.

69. [NEW] The computer readable medium of claim 68 wherein said computer-readable instructions stored thereon are such as to solicit user input which defines said presentation model in terms of an action selection presentation pattern which specifies in a hierarchical way what service presentation patterns, instance presentation patterns, class population presentation patterns and master-detail presentation patterns will be offered by the system as a means for users of said target computer program to interact with it.

70. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer, cause said computer to carry out the following process:

solicit user input which defines a presentation model which defines the user interface for a target computer program defined by a conceptual model, said solicited user input defining at least a service presentation pattern for every service of the system specifying how user of the system will be able to invoke a service of said system and said solicited user input further defining:

an optional introduction pattern assigned to every input data valuated argument of a service to which said service presentation pattern is assigned;

an optional defined selection pattern assigned to every input data

valuated argument of the service to which said service presentation pattern is assigned;

an optional population selection pattern assigned to every input object valuated argument of a service to which said service presentation pattern is assigned;

an optional dependency pattern assigned to every input argument of a service to which said service presentation pattern is assigned; and

converting said solicited user input into one or more data structures which implement said service presentation pattern.

71. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer, cause said computer to carry out the following process:

solicit user input which defines a presentation model which defines the user interface for a target computer program defined by a conceptual model, said solicited user input defining at least an instance presentation pattern for every class of the system specifying how users of said target computer program will be able to query the state of a given instance of a class of said target computer program, said solicited user input further comprising user input which defines:

a display set pattern;

a set of available services of the class owning said instance presentation pattern;

a set of navigations to presentation patterns owned by the classes related to a class owning said instance presentation pattern; and

converting said user input into one or more data structures implementing said instance presentation pattern.

72. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer, cause said computer to carry out the following process:

solicit user input which defines a presentation model which defines the user interface for a target computer program defined by a conceptual model, said solicited user input defining at least a class population presentation pattern for every class of the system specifying how users of said target computer program will be able to query the population of a given class of said target computer program, said solicited user input further comprising user input which defines:

a display set pattern;

an optional filter pattern;

an optional order criterion pattern;

a set of available services of the class owning said class population presentation pattern;

a set of navigations to presentation patterns owned by the classes related to a class owning said class population presentation pattern; and

converting said user input into one or more data structures implementing said class population presentation pattern.

73. [NEW] A computer-readable medium having stored thereon computer-readable instructions which, when executed by a computer, cause said computer to carry out the following process:

solicit user input which defines a presentation model which defines the user interface for a target computer program defined by a conceptual model, said solicited user input defining at least a master/detail presentation pattern for every class of the system specifying how users of said target computer program will be

able to query the state of a given instance of a class of said target computer program owning said master/detail presentation pattern (or the population of a class of said target computer program owning said master/detail presentation pattern), and the population of instances of classes related with an instance of the class of said target computer program owning said master/detail presentation pattern with said solicited user input further comprising user input which defines:

an instance presentation pattern, or a class population presentation pattern, defined for the class owning said master/detail presentation pattern, referred to as "master presentation pattern";

a set of presentation patterns which may be of type instance presentation pattern, class population presentation pattern, or master/detail presentation pattern, defined for classes related with the class owning the master/detail presentation pattern, referred to as "detail presentation patterns"; and

converting said user input into one or more data structures implementing said instance presentation pattern.

EVIDENCEAPPENDIX

None

RELATEDPROCEEDINGSAPPENDIX

None

Respectfully Submitted

Ronald C. Fish

Ronald Craig Fish
Reg. No. 28,843
Tel 408 866 4777
Attorney for Applicant(s)

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail, postage prepaid, in an envelope addressed to: Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450
on 5/11/06
(Date of Deposit)

Ronald C. Fish

Ronald Craig Fish, President
Ronald Craig Fish, a Law Corporation
Reg. No. 28,843